RNN唐诗生成报告

2253102 鲁浩宇

一.模型介绍

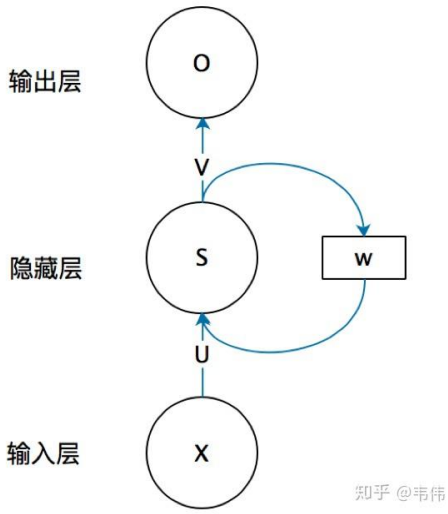
（1）RNN

概述

循环神经网络（RNN）是一类专门用于处理序列数据的神经网络架构。与传统的前馈神经网络不同，RNN具有"记忆"能力，能够利用先前时间步的信息来处理当前时间步的数据，这使得它在处理时间序列、自然语言、语音等序列数据时表现出色。

基本结构

RNN的核心思想是在网络中添加循环连接，使得信息能够在时间步之间传递。其基本结构可以表示为：



数学表示

对于一个简单RNN单元，在时间步t的计算可以表示为：

i. **隐藏状态更新**：

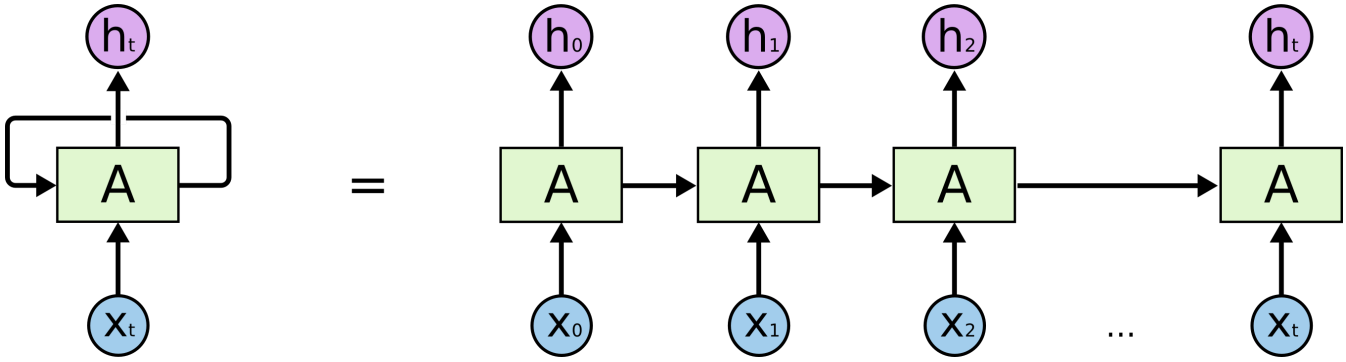
$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh} + b_h)$$

ii. **输出计算**：

$$y_t = W_{hy}h_t + b_y$$

展开形式

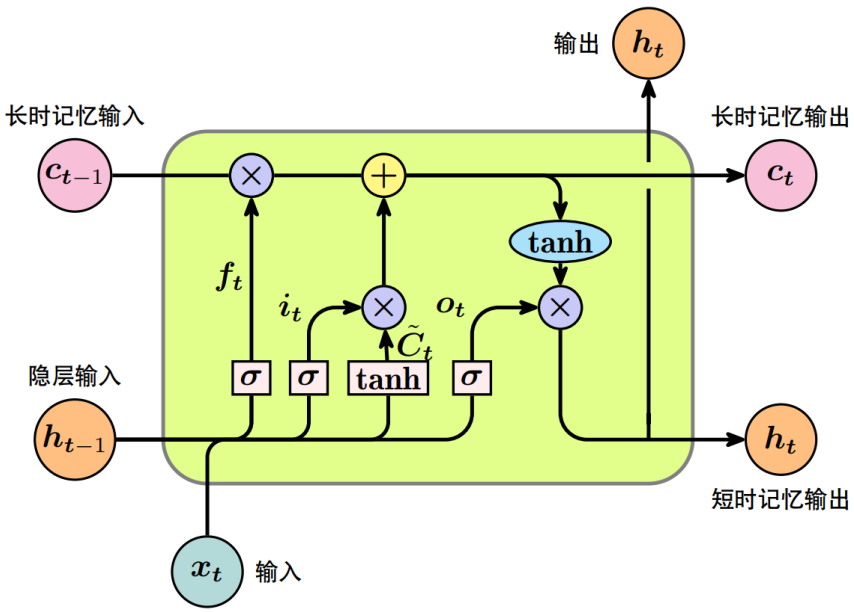
RNN可以按时间步展开，形成一个深度网络：



（2）LSTM

概述

LSTM是一种特殊的循环神经网络（RNN），由Hochreiter和Schmidhuber于1997年提出，用于解决传统RNN的梯度消失/爆炸问题。其核心思想是通过引入**门控机制**和**记忆单元**，实现对长期依赖关系的高效学习。LSTM在时间序列预测、自然语言处理（如机器翻译、文本生成）等领域表现优异。



LSTM的核心结构

LSTM通过三个门控单元（输入门、遗忘门、输出门）和一个记忆单元调控信息流动：

i. **遗忘门（Forget Gate）**
决定从记忆单元中丢弃哪些信息：

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

- σ：Sigmoid函数，输出值在[0,1]之间。

- W_f：权重矩阵，bf*bf*：偏置项。
- h_{t-1}：上一时刻的隐藏状态， x_t：当前输入。

ii. **输入门（Input Gate）**

控制新信息的更新：

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

$$C_t = tanh(W_C.[h_{t-1}, x_t] + b_C)$$

- C_t：候选记忆单元（临时状态）。

iii. **记忆单元更新（Cell State Update）**

结合遗忘门和输入门更新记忆单元：

$$C_t = f_t \odot C_{t-1} + i_t \odot C_t$$

- ⊙：逐元素乘法（Hadamard积）。

iv. **输出门（Output Gate）**

决定下一时刻的隐藏状态：

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$

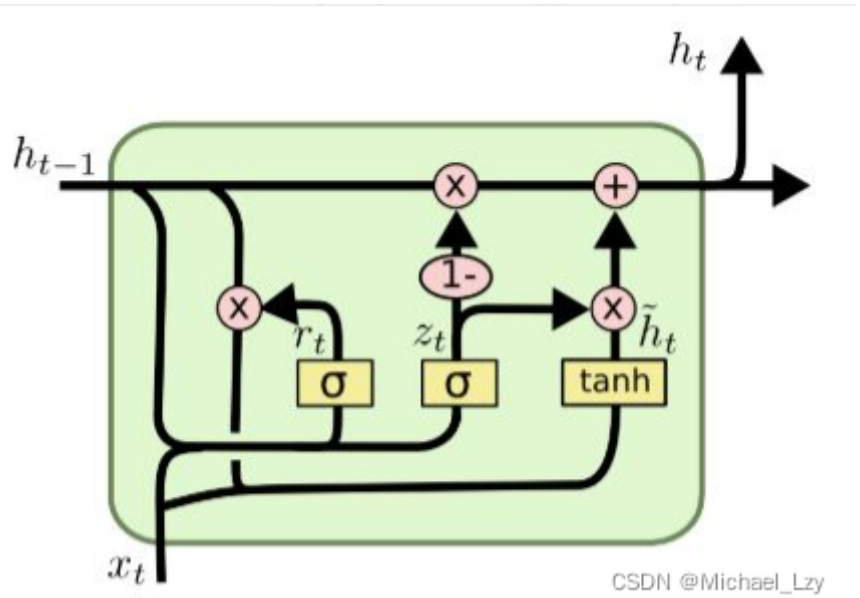
$$h_t = o_t \odot tanh(C_t)$$

- h_t：当前时刻的隐藏状态（输出）。

（3）GRU

概述

门控循环单元(Gated Recurrent Unit, GRU)是一种循环神经网络(RNN)的变体，由Cho等人在2014年提出。GRU通过引入门控机制来解决传统RNN中的梯度消失问题，同时相比LSTM具有更简单的结构。



基本结构

i. 更新门(Update Gate)

更新门决定有多少前一时刻的隐藏状态会被保留到当前时刻。

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

ii. 重置门(Reset Gate)

重置门决定有多少前一时刻的隐藏状态会被用于计算当前候选隐藏状态。

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

iii. 候选隐藏状态(Candidate Hidden State)

候选隐藏状态是基于重置门和当前输入计算得到的潜在新状态。

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b)$$

iv. 最终隐藏状态(Final Hidden State)

最终隐藏状态是前一时刻隐藏状态和候选隐藏状态的加权和，权重由更新门控制。

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

二.模型介绍

数据预处理

数据来源：系统从 `poems.txt` 文件中读取诗歌数据。

数据清洗：

- 去除诗歌中的空格、特殊符号（如'_', '(', '（', '《', '['等）。
- 过滤掉包含起始标记 `G` 或结束标记 `E` 的诗歌。
- 确保诗歌长度在5到80个字符之间。

标记处理：

- 每首诗歌前添加起始标记 `G` ，后添加结束标记 `E` 。

词汇表构建：

- 统计所有字符的出现频率，生成词汇表（ `word_int_map` ）和字符到索引的映射。
- 诗歌被转换为索引序列（ `poems_vector` ），便于模型处理。

模型架构

词嵌入层（ `word_embedding` ）：

- 将字符索引映射为稠密的词向量（维度为100）。

LSTM层：

- 使用两层LSTM，隐藏层维度为128。
- 支持批量处理（`batch_first=True`），并在训练时使用dropout（0.2）防止过拟合。

全连接层（`fc`）：

- 将LSTM的输出映射到词汇表大小的维度。

输出层：

- 使用`LogSoftmax`激活函数生成每个字符的概率分布。

训练过程

设备选择：自动检测并使用GPU（如果可用），否则使用CPU。

优化器与损失函数：

- 使用RMSprop优化器，学习率为0.01。
- 损失函数为负对数似然损失（`NLLLoss`）。

训练步骤：

- 将数据分批次输入模型。
- 计算损失并反向传播，使用梯度裁剪（`clip_grad_norm`）防止梯度爆炸。
- 每20个批次保存一次模型。
- 共训练30个epoch。

诗歌生成

输入处理：用户提供一个起始字符（如“日”）。

生成过程：

- 将起始字符转换为索引，输入模型。
- 模型逐步预测下一个字符，直到生成结束标记`E`或达到最大长度（30个字符）。
- 使用`to_word`函数将预测结果转换为字符。

输出格式化：通过`pretty_print_poem`函数去除起始和结束标记，并按句分行打印。

三.实验

训练过程

（采用GPU加速训练）

```
******************************
epoch   29 batch number 329 loss is:  5.9709062576293945
prediction [10, 5, 6, 20, 72, 0, 46, 686, 4, 54, 30, 1, 10, 27, 4, 74, 357, 0, 10, 179, 4, 26, 164, 1, 11, 104, 19, 32, 357, 0, 46, 228, 77, 31, 417, 1, 11, 134, 35, 114, 159, 0, 83, 16, 77, 83, 144, 1, 8,
 515, 9, 11, 126, 0, 46, 12, 22, 31, 246, 1, 3, 27, 4, 399, 104, 0, 4, 588, 92, 265, 5, 1, 3, 3]
b_y       [4149, 1307, 19, 984, 776, 0, 1476, 116, 95, 37, 5, 1, 4, 281, 804, 201, 48, 0, 617, 124, 40, 118, 13, 1, 479, 104, 2229, 52, 357, 0, 2788, 228, 89, 640, 362, 1, 555, 129, 1154, 206, 73, 0, 88,
16, 676, 125, 76, 1, 1166, 1166, 1349, 1356, 45, 0, 75, 743, 951, 1110, 121, 1, 103, 180, 729, 613, 1663, 0, 4, 348, 175, 731, 535, 1, 3, 3]
```

模型效果

以“日”为开头的诗

日日西东风。
风吹水，水不通，山河风前不可见。
不知一片云，一片青。

以“红”为开头的诗

红颜笑酒行。
风光生不得，风雨自无风。
风雨云犹在，风吹水自浮。

以“山”为开头的诗

山中见，清风吹玉台。
风光一片片，万里无人期。
风雨不可见，风吹不。

以“夜”为开头的诗

夜长生，清风吹玉台。
风光不可见，风雨自无风。
不是无人见，何人更。

以“湖”为开头的诗

湖中，高树见人间。
北望千门里，千山万里行。
风光生雨后，山水暮潮。

以“海”为开头的诗

海上客，东风吹不回。
山川无处处，风雨不可闻。
风光满山水，山水见。

以“月”为开头的诗

月明秋风生碧水。
一夜一相望，一生千万人。

四.总结

本实验基于RNN、LSTM和GRU构建了古诗生成模型，通过处理诗歌数据集（`poems.txt`）并添加起始/结束标记，使用双层LSTM（隐藏层128维）结合词嵌入（100维）和Dropout（0.2）进行训练。模型采用RMSprop优化器和NLLLoss损失函数，支持GPU加速与梯度裁剪。生成诗歌时，用户输入起始字符（如“日”），模型逐步预测直至结束标记，输出格式清晰。实验表明，模型能生成结构完整、意境连贯的五言或七言诗，验证了LSTM在序列生成任务中的有效性，但部分诗句的创造性仍有提升空间。未来可通过扩大数据集或引入注意力机制进一步优化。